INGENIEUR EN SCIENCES INFORMATIQUES ET
MASTER INFORMATIQUE FONDEMENTS ET INGENIERIE

RAPPORT DU PROJET WEB

Etudiant : Florian Ainadou (SI5- Parcours AL)

Responsables : Peter Sanders, François Michaudon, Hugo Mallet

PLAN DU RAPPORT

Identifiant GitHub

FlorianAinadou

Tâches effectuées

- Localisation de l'utilisateur sur la Carte

- Chargement des périodes sur le graphe et amélioration du graphe utilisé

Stratégie employée pour la gestion des versions avec Git

Pour ce qui est de la gestion des versions, nous avons utilisé GitHub. Nous avions au départ un git pour la partie Backend de l'application. Par la suite, nous y avons rajouté la partie Frontend. Pour le développement en lui-même, nous avons utilisé un système de branches. Nous avions deux branches principales : **develop** et **main**. Au départ nous étions partis en développement des fonctionnalités du backend sur la branche **back**. Par suite de l'intégration de la partie Frontend, nous avons commencé à intégrer les fonctionnalités relatives au Frontend sur une branche **front.**

La branche **main** reste la branche principale et **develop** la branche pour intégrer les fonctionnalités avant de les pousser vers **main.** Pour certaines fonctionnalités, il a été nécessaire de les développer sur d'autres branches afin de pouvoir faire des tests sereinement. Par la suite, nous avons utilisé la branche **lab.** pour merger les différentes fonctionnalités des branches **back** et **front** pour nous assurer que **le tout était bien fonctionnel**. Et pour finir, nous avons poussé les fonctionnalités de cette branche vers la branche **main**.

Nous avons aussi utilisé des tickets pour avoir la possibilité de suivre la cohérence des fichiers comités avec les taches auxquelles il fait référence. Nous aurions pu ajouter un système de « **Milestone »** afin de mieux regrouper les taches qui allait dans le sens d'une même fonctionnalité mais en cours de développement, nous n'en avons pas vu l'utilité.

Solutions choisies

Nous avons utilisé l'API HTML5 pour la localisation et Leaflet pour la carte. Nous avons utilisé Leaflet pour plusieurs raisons notamment son grand nombre de téléchargement. En effet, vu les temps impartis, il nous fallait trouver une librairie utilisée de manière fréquente afin d'être surs d'avoir le support nécessaire lors du développement et en cas d'erreurs.

Pour les données, nous avons récupérées celles qu'on retrouve sur le site du gouvernement. Nous récupérons ces données au format CSV puis les utilisons pour populer la base de données MongoDB. Nous aurions pu utiliser des API disponibles en ligne, mais nous avons choisi d'utiliser les données fournies par le Backend pour montrer notre capacité à lier les deux projets.

Pour les graphes, nous avions commencé le développement avec '**react-apexcharts**' qui n'était cependant pas notre choix initial. Je le présenterai un peu plus bas

Difficultés rencontrées

La principale difficulté lors de ce projet a été de s'adapter au linter. Nous avions beaucoup d'erreurs qui nous ont fait perdre pas mal de temps dans le développement. Il nous a fallu chercher un bon moment pour comprendre les réelles raisons de nos erreurs, mais au moins, nous avons été forcés de respecter un bon nombre de règles. Ensuite, les **hooks** nous ont aussi poser quelques problèmes. Dans le but de factoriser au mieux le code, nous avons créer plusieurs composants et plusieurs fonctions utilitaires. Mais la mauvaise maîtrise des règles de React nous a rendu la tâche ardue. Pour l'utilisation de la localisation sur la carte, Il était plus ou moins compliqué de gérer correctement l'état de la page et des données qui y étaient intégrées.

Pour les graphes, nous avions commencé le développement avec **'react-chartjs-2'** qui proposait une API simple pour la réalisation des graphes. Mais par la suite, elle n'exposait pas vraiment de solution pour l'affichage de périodes sur le graphe. Nous avons donc été amenés à changer de solution et à nous pencher sur '**react-apexcharts**' qui nous proposait une solution plus ou moins complète et bien documentée.

Temps de développement / tâche

- Localisation de l'utilisateur sur la Carte : Environs 3 h

- Chargement des périodes sur le graphe et amélioration du graphe utilisé : environs 2 h

Code

Composant optimal (Location.js : https://github.com/wak-nda/prog-web-ADKN/blob/main/frontend/src/services/Location.js)

Il s'agit là d'un **hook** que j'ai écrit pour la gestion de la localisation. Il permet de qui utilise les hooks **useState** et **useEffect** pour afin de l'utiliser plus facilement sur la carte. Il a vu le jour à cause de la difficulté à gérer facilement les états sur la carte. Malheureusement, il ne sera pas utilisé au final pour des raisons techniques.

Composant à faire évoluer (ChartsRegions.js : https://github.com/wak-nda/prog-web-ADKN/blob/main/frontend/src/components/ChartsRegions.js)

Ce composant est utilisé pour afficher les graphes. Il y a beaucoup de duplication de code pour des raisons de temps. J'aurais pu le gérer autrement en créant un composant que je pourrais répéter plusieurs fois et en utilisant les **props** pour passer les données nécessaires.